When users interact with a server, either exposed to the Internet or within an intranet, **privacy issues**
arise. They become extremely worrying in public clouds, where data are provided to an infrastructure hosted outside user's premises. The right to
act without observation
becomes even more important in Desktop-as-a-Service (DaaS) environments. When somebody stores personal information on her account at a DaaS like VDD, she would be sure that her data would remain safe and nobody but the owner can read them. It would be also desirable to make the whole list of operations performed by a user on her account, obscure or meaningless for the system administrator.
At the moment, VDD project is focusing on how to face such a problem. We have performed a preliminary experimental evaluation of a Progressive Privacy solution for a DaaS system. **Prog ressive Privacy**
is a privacy preserving model which can be configurable (possibly on-demand) by a user not only quantitatively but rather qualitatively, i.e., the user is allowed to discriminate what type of information must be preserved and to what extent, according to her desired profiles of privacy. To this end, a lightweight client-side proxy named
Hedge Proxy
has been designed such that non-intelligible user contents and non-traceable user actions are guaranteed by enabling
Homomorphic Encryption
,
Oblivious Transfer
and
Query Obfuscation
schemes in the proxy.

Homomorphic Encryption is an encryption scheme based on algebraic properties of encrypted messages, i.e. being operands of functions returning coherent encrypted results. Consider the following scenario. A user owns a message *x* she wishes to keep private while copying it at a service provider space. Traditional encryption schemes are sufficient to meet the privacy requirement. However, with traditional schemes if the user wishes to perform some function
f
on

x
, a decryption of
x
server-side is required at some point, thus revealing
x
to the service provider and violating user's privacy. With homomorphic encryption it is possible to copy
x
in an encrypted form at the service provider space; the user performs any
f
on it and obtains a result which is the encrypted value of
f(x)
. In this way, the user can let the service provider work on her data that are kept private, as the service provider has no means to read
x
.

Once the data are protected using homomorphic encryption schemes, it may be required that also the function *f* representing a computation to be performed on the encrypted content is to be maintained private. In **Oblivious Transfer**, one party, the sender, transmits some information to another party, the chooser, in a manner that protects both of them: the chooser is assured that the sender does not learn which part of the information it is received, while the sender is assured that the chooser does not receive more information than that it is entitled to. In the context of privacy preserving in DaaS, the latter guarantee can be relaxed since all the information is owned by the chooser (the cloud client) or openly offered to the chooser (i.e. applications of the Cloud system). In DaaS, only the first oblivious transfer guarantee is relevant; that is, the provider does not learn what piece of data is requested and what function is requested to be executed on it.

Query Obfuscation is another privacy preserving scheme that can be successfully used in case it is required to guarantee privacy of data computations. The scheme consists in introducing noisy or fake queries at random or mixing queries from different users so that cloud providers are not able to know which actions are executing on the data.

Preliminary results of Hedge Proxy evaluation are used to assess the performances experienced by users of VDD against the progressive privacy achievements that can be obtained. As expected, the perceived client performances when using VDD highly decrease when augmenting the level of privacy protection (e.g., using large key encryption size, high obfuscation density). Nevertheless, experiments show that the system can reach **fair level of privacy of a light data stream encrypted using small keys without significantly deteriorating user experienced performances**
; that is, they show that the above mentioned cryptographic schemes can be practically used even in the cloud computing context, despite their high communication and computation costs.

Currently, a massive series of experiments are being carried out showing which is the latency experienced by a user that is running common applications on DaaS, such as mailer programs,

browsers etc, when increasing data stream size and/or protection level. These results will be presented when ready; however the communication and computation costs would suggest to split users in separate classes where each user in a certain class can exploit the advantages of progressive privacy using different key sizes and obfuscation levels according to their status.